Writeup Cyber Jawara 2017

Neither Fast Nor Accurate

CrackMe (RE)

File:

```
$ curl -Ls https://git.io/vdqDo | base64 -d > numbers
```

Diberikan sebuah binary 64 bit yang meminta inputan passcode, terlihat pada line 25 bahwa panjang passcode yaitu 34. Pada pengecekan pertama akan di XOR kiri dan kanan yang hasilnya akan dibandingkan dengan nilai di db dword_201020. Selanjutnya bagian kiri di XOR dengan 0x45 yang hasilnya akan dibandingkan dengan db dword_201080. Berikut hasil decompilenya.

```
1  __int64 sub_850()
2  {
3    int v1; // [sp+4h] [bp-43Ch]@1
4    signed int i; // [sp+4h] [bp-43Ch]@8
5    int v3; // [sp+8h] [bp-438h]@1
6    signed int v4; // [sp+Ch] [bp-434h]@1
7    char dest[17]; // [sp+10h] [bp-430h]@8
8    char v6; // [sp+21h] [bp-41Fh]@8
9    char s[1032]; // [sp+30h] [bp-410h]@1
10   __int64 v8; // [sp+438h] [bp-8h]@1
11
12   v8 = *MK_FP(__FS__, 40LL);
13   printf("Insert Passcode: ");
14   __isoc99_scanf("%s", s);
15   v1 = 0;
16   v3 = strlen(s) - 1;
17   v4 = 1;
18   while ( v1 <= v3 )
19   {
20     if ( (char)(s[v1] ^ s[v3]) != dword_201020[v1] )
21       v4 = 0;
22     ++v1;
23     --v3;
24   }
25   if ( strlen(s) == 34 && v4 )
26   {
27     memcpy(dest, s, 0x11uLL);
28     v6 = 0;
29     for ( i = 0; i <= 16; ++i )
30     {
31       if ( (char)(dest[i] ^ 0x45) != dword_201080[i] )
32         v4 = 0;
33     }
34     if ( v4 )
35     {
36       puts("Correct!");
37       puts(s);
38     }
39     else
40     {
41       puts("Incorrect");
42     }
43   }
44   else
45   {
46     puts("Incorrect");
47   }
48   return *MK_FP(__FS__, 40LL) ^ v8;
49 }
```

Pertama kita akan buat bagian kiri terlebih dahulu karena lebih mudah.

```
#!/usr/bin/env python

db2 = [6, 0xF, 0x77, 0x75, 0x74, 0x72, 0x3E, 0x22, 0x75, 0x75, 0x21,0x1A, 0x29, 0x30, 0x26,
0x2E, 0x1A]
# bagian kiri

left = [d ^ 0x45 for d in db2]
```

Selanjutnya bagian kanan, nilai yang ada pada list ‘left’ akan di XOR dengan nilai di db1 untuk mendapatkan nilai bagian kanan, karena urutannya terbalik maka listnya dibalik dengan [::-1]

```
# bagian kanan
db1 = [0x3E, 0x6B, 0x5E, 0x5C, 0x50, 0x68, 0x15, 0x12, 0x58, 0x40, 0x3B, 0x3A, 0x1A, 0x14,
0x0B, 0x34, 0x79]

db1 = [0x3E, 0x6B, 0x5E, 0x5C, 0x50, 0x68, 0x15, 0x12, 0x58, 0x40, 0x3B, 0x3A, 0x1A, 0x14,
0x0B, 0x34, 0x79]
right = [d ^ left[i] for i, d in enumerate(db1)][::-1]
flag = left + right

print"".join([chr(f) for f in flag])

# CJ2017{g00d_luck_&_have_phun_all!}
```

Numbers (RE)

File:

```
$ curl -Ls https://git.io/vdqDo | base64 -d > numbers
```

Diberikan sebuah file stripped binary 64 bit, program tersebut meminta 6 angka lalu diolah di suatu fungsi, hasil dari fungsi tersebut akan dibandingkan dengan angka 98561, 1507861, 180346, 22005, 120871, dan 1311561. Untuk memudahkan kita sebut saja fungsinya sebagai olah_int. Berikut hasil decompilernya.

```
_isoc99_scanf("%d %d %d %d %d %d", &v11, &v12, &v13, &v14, &v15);
LODWORD(v3) = sub_7B0(v11);
if ( v3 != 98561
  || (LODWORD(v4) = sub_7B0(v12), v4 != 1507861)
  || (LODWORD(v5) = sub_7B0(v13), v5 != 180346)
  || (LODWORD(v6) = sub_7B0(v14), v6 != 22005)
  || (LODWORD(v7) = sub_7B0(v15), v7 != 120871)
  || (LODWORD(v8) = sub_7B0(v16), v8 != 1311564) )
{
  puts("Wrong!");
}
else
{
  puts("Right!");
  printf("CJ2017{%d-%d-%d-%d-%d-%d}\n", v11, v12, v13, v14, v15, v16);
}
result = 0LL;
```

Sebenarnya pada saat lomba kami menyelesaikannya dengan mengimplementasi ulang fungsi olah_int dengan C lalu di bruteforce namun pada writeup ini akan kami coba menggunakan Frida. Frida[1] adalah sebuah framework reverse engineering dynamic instrumentation, frida dapat hook fungsi pada program yang sedang berjalan. Sebelum kita hook fungsi sub_7B0 di program terlebih dahulu kita harus tau alamat fungsi tersebut. Namun sayangnya binary yang dikasih terdapat proteksi PIE[2] (Position Independent Executable) sehingga alamat fungsi di region .text ikut teracak oleh ASLR (Address Space Layout Randomization). Jadi kita tidak bisa langsung hardcode alamat olah_int di script, harus leak base address dulu menggunakan frida yang kemudian dijumlah dengan offset alamat olah_int.

[1] https://www.frida.re/
[2] https://en.wikipedia.org/wiki/Position-independent_code

```
.text:00000000000007B0
.text:00000000000007B0 sub_7B0         proc near               ; CODE XREF: sub_7B0+2D↓p
.text:00000000000007B0                                         ; sub_7B0+66↓p ...
.text:00000000000007B0
.text:00000000000007B0
.text:00000000000007B0 var_4           = dword ptr -4
.text:00000000000007B0
.text:00000000000007B0                 push    rbp
.text:00000000000007B1                 mov     rbp, rsp
.text:00000000000007B4                 sub     rsp, 10h
.text:00000000000007B8                 mov     [rbp+var_4], edi
.text:00000000000007BB                 cmp     [rbp+var_4], 1
.text:00000000000007BF                 jg      short loc_7CB
.text:00000000000007C1                 mov     eax, 1
.text:00000000000007C6                 jmp     locret_8B0
.text:00000000000007CB ; ---------------------------------------------------------------------------
.text:00000000000007CB
```

Didapatkan offset 0x7B0. Base address bisa didapatkan dengan api Frida enumerate_modules().

```
#!/usr/bin/env python

import frida
import sys

def main():
    prog = "numbers"
    # attach frida pada proses yang lagi berjalan
    session = frida.attach(prog)
    # dapatkan base address
    for module in session.enumerate_modules():
        if module.name == prog:
            base = module.base_address
            break
    print("base address: %s" % hex(base))
    offset = 0x7b0
    olah_int = base + offset
    print("olah_int address: %s" % hex(olah_int))
```

Selanjutnya buat script js untuk bruteforce

```
   script_src = """
    check_number = new NativeFunction(ptr("%s"), 'int', ['int']);
    win_db = [98561, 1507861, 180346, 22005, 120871, 1311564];
    solution = [];
    setTimeout(function () {
        for(i=0; i < win_db.length; i++){
            for(j=0; j<0xffffffff; j++){
                if(check_number(j) == win_db[i]){
                    console.log("index", i, "cracked");
                    solution.push(j);
                    break;
                }
            }
        }
        console.log(solution.join(" "));
    }, 0);
    """
    script = session.create_script(script_src % olah_int)
```

```
    # load script
    script.load()
```

Berikut script lengkap Frida untuk bruteforce soal numbers

```
#!/usr/bin/env python

import frida
import sys

def main():
    prog = "numbers"
    # attach frida pada proses yang lagi berjalan
    session = frida.attach(prog)
    # dapatkan base address
    for module in session.enumerate_modules():
        if module.name == prog:
            base = module.base_address
            break
    print("base address: %s" % hex(base))
    offset = 0x7b0
    olah_int = base + offset
    print("olah_int address: %s" % hex(olah_int))
    script_src = """
    check_number = new NativeFunction(ptr("%s"), 'int', ['int']);
    win_db = [98561, 1507861, 180346, 22005, 120871, 1311564];
    solution = [];
    setTimeout(function () {
        for(i=0; i < win_db.length; i++){
            for(j=0; j<0xffffffff; j++){
                if(check_number(j) == win_db[i]){
                    console.log("index", i, "cracked");
                    solution.push(j);
                    break;
                }
            }
        }
        console.log(solution.join(" "));
    }, 0);
    """
    script = session.create_script(script_src % olah_int)

    # load script
    script.load()

    try:
        while True:
            pass
    except KeyboardInterrupt:
        session.detach()
        sys.exit(0)

if __name__ == '__main__':
    main()
```

Untuk melihat scriptnya berjalan dapat ditonton di asciinema[3] tidak sampai 1 menit flag didapatkan.

[3] https://asciinema.org/a/vZpp8p6vPKHh4NnnqjQFgbDaq

```
$ ./numbers
5981 91457 10934 1337 7331 79548
Right!
CJ2017{5981-91457-10934-1337-7331-79548}
```

Game (RE)

File:

```
$ curl -Ls https://git.io/vdZSY | base64 -d > game
```

Diberikan sebuah binary yang merupakan game. Pertama-tama kita terkunci disuatu ruangan, terdapat pintu namun kita diminta untuk menemukan kunci terlebih dahulu.

Terdapat string output “You found the key!” namun kami tidak tahu kapan fungsi itu dipanggil, setelah di dereferensi, sepertinya switch case di C sulit dipahami

```
.text:00000000000018C9                mov     rax, [rbp+var_18]
.text:00000000000018CD                mov     rdi, rax
.text:00000000000018D0                call    showmap
.text:00000000000018D5                lea     rdi, aGame      ; "\n[GAME]"
.text:00000000000018DC                call    puts
.text:00000000000018E1                lea     rdi, aHHeroStats ; "(h) Hero Stats"
.text:00000000000018E8                call    puts
.text:00000000000018ED                lea     rdi, aLLegends  ; "(l) Legends"
.text:00000000000018F4                call    puts
.text:00000000000018F9                lea     rdi, aWMoveUp   ; "(w) Move Up"
.text:0000000000001900                call    puts
.text:0000000000001905                lea     rdi, aDMoveRight ; "(d) Move Right"
.text:000000000000190C                call    puts
.text:0000000000001911                lea     rdi, aSMoveDown ; "(s) Move Down"
.text:0000000000001918                call    puts
.text:000000000000191D                lea     rdi, aAMoveLeft ; "(a) Move Left"
.text:0000000000001924                call    puts
.text:0000000000001929                lea     rdi, a0Exit     ; "(0) Exit"
.text:0000000000001930                call    puts
.text:0000000000001935                mov     edi, 0Ah        ; c
.text:000000000000193A                call    putchar
.text:000000000000193F                lea     rdi, aYourChoice_0 ; "Your Choice: "
.text:0000000000001946                mov     eax, 0
.text:000000000000194B                call    printf
.text:0000000000001950                lea     rax, [rbp+var_9]
.text:0000000000001954                mov     rsi, rax
.text:0000000000001957                lea     rdi, aC         ; "\n%c"
.text:000000000000195E                mov     eax, 0
.text:0000000000001963                call    __isoc99_scanf
.text:0000000000001968                movzx   eax, [rbp+var_9]
.text:000000000000196C                movsx   eax, al
.text:000000000000196F                sub     eax, 30h
.text:0000000000001972                cmp     eax, 47h
.text:0000000000001975                ja      loc_1A39
.text:000000000000197B                mov     eax, eax
.text:000000000000197D                lea     rdx, ds:0[rax*4]
.text:0000000000001985                lea     rax, dword_2014
.text:000000000000198C                mov     eax, [rdx+rax]
.text:000000000000198F                movsxd  rdx, eax
.text:0000000000001992                lea     rax, dword_2014
.text:0000000000001999                add     rax, rdx
.text:000000000000199C                jmp     rax
```

Ketika didecompile hasilnya hanya seperti ini:

```
  6
  7  v4 = *MK_FP(__FS__, 40LL);
  8  putchar(10);
  9  showmap(a1);
 10  puts("\n[GAME]");
 11  puts("(h) Hero Stats");
 12  puts("(l) Legends");
 13  puts("(w) Move Up");
 14  puts("(d) Move Right");
 15  puts("(s) Move Down");
 16  puts("(a) Move Left");
 17  puts("(0) Exit");
 18  putchar(10);
 19  printf("Your Choice: ");
 20  _isoc99_scanf("\n%c", &v3);
 21  v1 = v3 - 48;
 22  if ( v1 <= 'G' )
 23    JUMPOUT(__CS__, dword_2014 + dword_2014[v1]);
 24  return *MK_FP(__FS__, 40LL) ^ v4;
 25}
```

Inputan kita haruslah <= ‘G’ + 48 atau <= 119 agar program melakukan jump. Karena kita tidak tahu command mana yang akan jump ke fungsi key maka kita akan bruteforce dengan gdb scripting. Pasang breakpoint di ‘jmp rax’. Untuk debug binary dengan PIE sebaiknya matikan ASLR agar base address PIE selalu pada 0x555555...4000

```
.text:000000000000196F                sub     eax, 30h
.text:0000000000001972                cmp     eax, 47h
.text:0000000000001975                ja      loc_1A39
.text:000000000000197B                mov     eax, eax
.text:000000000000197D                lea     rdx, ds:0[rax*4]
.text:0000000000001985                lea     rax, dword_2014
.text:000000000000198C                mov     eax, [rdx+rax]
.text:000000000000198F                movsxd  rdx, eax
.text:0000000000001992                lea     rax, dword_2014
.text:0000000000001999                add     rax, rdx
.text:000000000000199C                jmp     rax
.text:000000000000199E ; ---------------------------------------------------------------
```

```
import gdb
import string

guess = string.ascii_letters + string.digits
jmp_addr = 0x199c

def clean_reg(_str):
    return int(_str.split("\t")[1].rstrip('\n'), 16)

gdb.execute('file game')
gdb.execute('set pagination off')
gdb.execute('b *0x555555554000 + %s' % (str(jmp_addr)))

rax_dict = []

for g in guess:
        if ord(g) <= 119:
                gdb.execute('r < <(echo {})'.format(g), True, True)
                RAX = clean_reg(gdb.execute('i r rax', True, True))
                if RAX != 41520074072490295:
                        rax_dict.append(g)
        else:
                continue
print(rax_dict)

# ['a', 'd', 'h', 'k', 'l', 's', 'w', '0']
```

Ternyata ada command yang tidak ada di petunjuk yaitu ‘k’, kita coba masukan ‘k’ di input

```
(a) Move Left
(0) Exit

Your Choice: k

Breakpoint 1, 0x000055555555599c in option ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[-------------------------------------------------------------------------------------------------------------REGISTERS-------------------------------------------------------------------------------------------------------------]
*RAX  0x555555555a2b (option+391) ← mov    rax, qword ptr [rbp - 0x18]
 RBX  0x0
*RCX  0x7ffff7b94b58 ← 0xffed94b6ffed93d5
*RDX  0xffffffffffffa17
*RDI  0x7fffffffd810 ← 0x4
*RSI  0x1
 R8   0x0
*R9   0xffffffffffffff98
*R10  0x6b
*R11  0x246
*R12  0x555555554a10 (_start) ← xor    ebp, ebp
*R13  0x7fffffffde70 ← 0x1
 R14  0x0
 R15  0x0
*RBP  0x7fffffffdd50 → 0x7fffffffdd70 → 0x7fffffffdd90 → 0x555555555b10 (__libc_csu_init) ← ...
*RSP  0x7fffffffdd30 ← 0x2323232323232323 ('########')
*RIP  0x555555555599c (option+248) ← jmp    rax
[-------------------------------------------------------------------------------------------------------------DISASM-------------------------------------------------------------------------------------------------------------]
 ► 0x555555555599c <option+248>    jmp    rax                          <0x555555555a2b>
    ↓
   0x555555555a2b <option+391>    mov    rax, qword ptr [rbp - 0x18]
   0x555555555a2f <option+395>    mov    rdi, rax
   0x555555555a32 <option+398>    call   key                          <0x55555555585d>

   0x555555555a37 <option+403>    jmp    option+406                   <0x555555555a3a>
    ↓
```

Terlihat RAX menuju 0x555555555a2b yaitu case ke pengecekan key artinya command ‘k’ untuk pengecekan key. Setelah dilihat hasil decompile fungsi key

```
1  int __fastcall key(__int64 a1)
2  {
3    int result; // eax@1
4
5    result = *(a1 + 20);
6    if ( result == 5 )
7    {
8      result = *(a1 + 24);
9      if ( result == 18 )
10     {
11       result = keyfound;
12       if ( !keyfound )
13       {
14         keyfound = 1;
15         result = puts("You found the key!");
16       }
17     }
18   }
19   return result;
20 }
```

‘H’ harus berada pada kordinat 5, 18 dari map yaitu posisi pojok kanan bawah untuk mendapatkan kunci.

```
####################
####............##.#
###....#..#.#....#.#
##.B..###......#...#
#####...#.....#.##.#
```

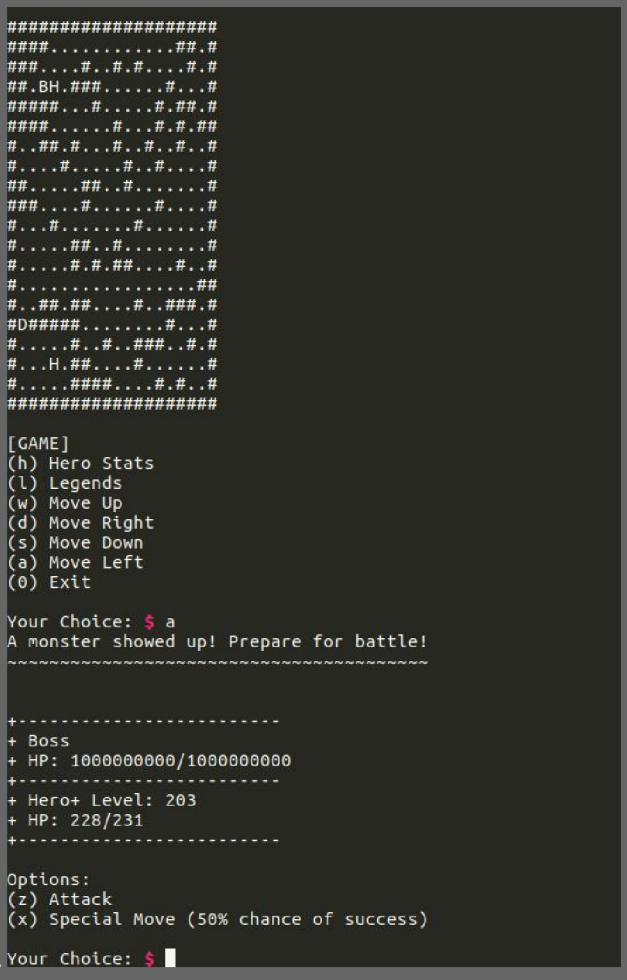
```
####......#...#.#.##
#..##.#...#..#..#..#
#....#.....#..#....#
##.....##..#.......#
###....#......#....#
#...#.......#......#
#.....##..#........#
#.....#.#.##....#..#
#..................##
#..##.##....#..###.#
#D#####........#...#
#.....#..#..###..#.#
#...H.##....#......#
#....H####....#.#..#
####################

[GAME]
(h) Hero Stats
(l) Legends
(w) Move Up
(d) Move Right
(s) Move Down
(a) Move Left
(0) Exit

Your Choice: k
You found the key!
```

Setelah berhasil keluar dari pintu kita diminta berburu monster agar levelnya naik, untuk memudahkan level naik kami buat script otomatisnya

```
#!/usr/bin/env python

from pwn import *

p = process('./game')

def action(act):
        log.info("send: %s", act)
        for a in act:
                p.sendline(a)
        resp = p.recv()
        log.info(resp)
        return resp

def get_key():
        action("sd") # pojok
        action("k")

def door():
        action("wwaaaawwd")

def beat():
        resp = action('x')
        while "Won!" not in resp:
                resp = action('x')
        action('3')

def levelup(num):
        level = 1
        move = 'w'
```

```
        while level != num:
                resp = action(move)
                if "wall" in resp and move == 'w':
                        move = 's'
                elif "wall" in resp and move == 's':
                        move = 'w'

                if "monster" in resp:
                        beat()
                        level += 1

        p.interactive()

def main():
        get_key()
        door()

        levelup(200)

if __name__ == "__main__":
        main()
```

Setelah menunggu lama hingga level 200 kami mencoba melawan bos yang ada di ‘B’, ternyata kekuatan bos jauh melampaui kekuatan hero



Akhirnya kami mencoba melihat cara alternatif lain, terdapat fungsi mencurigakan di fungsi action

```
if ( *level > 5 )
{
  v8 = 0;
  for ( i = pos - 1; i >= 0; --i )
  {
    v6 = convert(hist_r[i], hist_c[i]);
    v2 = v8++;
    if ( tt[v2] != v6 )
      return *MK_FP(__FS__, 40LL) ^ v14;
    v3 = v8;
    if ( v3 >= strlen(tt) )
      break;
  }
  puts("Cheat Activated");
  *(level + 12) = 2000000000;
  *(level + 4) = 2000000000;
  *(level + 16) = 2000000000;
  *(level + 8) = 2000000000;
}
return *MK_FP(__FS__, 40LL) ^ v14;
```

Terdapat hist_r dan hist_c , setelah dianalisa setiap pergerakan berikut perubahan nilainya

	hist_r	hist_c
a	0	-1
w	-1	0
s	1	0
d	0	1

Terdapat fungsi convert yang merubah gerakan menjadi simbol

```
s =>'%'
w => '&'
d => '$'
a => '^'
```

Aktivasi cheat akan terjadi jika gerakannya sesuai dengan

```
%%%&&&$$$$$$$$%%^^&&$$%%$$^$^$
```

Atau

```
sssswwwddddddddssaawwddssddadad
```

Jika dikonversi ke simbol gerak.

Karena pengecekan dimulai dari gerakan terakhir maka kita harus melakukan gerakan dibalik

```
Gerakan:
dadaddssddwwaassddddddddwwwssss

Map pergerakan:
         x
Hxxxx    x
  x x    x
  xxxxxxxx
         x
```

Pada kordinat mana kita dapat melakukan gerakan itu? Jawabannya ada di kordinat (8, 9)

Berikut scriptnya

```
#!/usr/bin/env python

from pwn import *

p = process('./game')
```

```
def action(act):
	log.info("send: %s", act)
	for a in act:
		p.sendline(a)
	resp = p.recv()
	log.info(resp)
	return resp

def get_key():
	action("sd") # pojok
	action("k")

def door():
	action("wwaaaawwd")

def beat():
	resp = action('x')
	while "Won!" not in resp:
		resp = action('x')
	action('3')

def levelup(num):
	level = 1
	move = 'w'
	while level != num:
		resp = action(move)
		if move == 'w':
			move = 's'
		elif move == 's':
			move = 'w'

		if "monster" in resp:
			beat()
			level += 1

def goto_cheat():
	moves = "ddddddwwww"
	for m in moves:
		resp = action(m)
		if "monster" in resp:
			beat()

def cheat():
	patt = "dadaddssddwwaassddddddwwwssss"
	for p in patt:
		resp = action(p)
		if "monster" in resp:
			beat()

def main():
	get_key()
	door()
	levelup(6)
	goto_cheat()
	cheat()


	p.interactive()

if __name__ == "__main__":
	main()
```

```
(3) Max HP

Your Choice: $ 3

----- Hero Stats -----
Level: 13
HP: 42/42
Attack: 5
Defense: 5
----------------------
Cheat Activated

####################
####............##.#
###....#..#.#....#.#
##.B..###......#...#
#####...#.....#.##.#
####......#...#.#.##
#..##.#...#..#..#..#
#....#.....#..#....#
##.....##..#.......#
###....#......#....#
#...#.......#......#
#.....##..#........#
#.....#.#.##....#.H#
```

Sekarang tinggal menuju Boss untuk dikalahkan

```
[*] Switching to interactive mode
$ h

----- Hero Stats -----
Level: 8
HP: 2000000000/2000000000
Attack: 2000000000
Defense: 2000000000
----------------------
```

Keygen Redux (RE)

File:

```
$ curl -Ls https://git.io/vdY0B | base64 -d > keygen_redux
```

Diberikan sebuah program yang meminta 10 serial number yang valid dan unik. Jika serial number yang valid dimasukkan kembali maka program akan mengembalikan “Duplicate”

```
v8 = *MK_FP(__FS__, 40LL);
v6 = 0;
puts("Insert 10 valid Serial Number:");
v0 = &byte_1187;
puts(&byte_1187);
for ( i = 0; i <= 9; ++i )
{
  _isoc99_scanf("%s", &s);
  if ( sub_DE6(&s, v6) )
  {
    puts("Duplicate");
    exit(0);
  }
  if ( !sub_A59(&s) )
  {
    puts("Invalid Serial Number");
    exit(0);
  }
  puts("OK");
  v1 = strlen(&s);
  qword_202040[v6] = (char *)malloc(v1 + 5);
  v2 = strlen(&s);
  v3 = v6++;
  strncpy(qword_202040[v3], &s, v2);
  puts(&byte_1187);
  v0 = &byte_1187;
  puts(&byte_1187);
}
```

Terdapat fungsi pengecekan sub_A59 setelah dilihat hasil decompilenya

```
1  __int64 __fastcall sub_A59(__int64 a1)
2  {
3    __int64 result; // rax@2
4    signed int v2; // [sp+1Ch] [bp-24h]@3
5    signed int v3; // [sp+20h] [bp-20h]@16
6    int i; // [sp+24h] [bp-1Ch]@16
7    int k; // [sp+24h] [bp-1Ch]@26
8    int j; // [sp+28h] [bp-18h]@17
9    int v7; // [sp+20h] [bp-10h]@26
10
11   if ( strlen(a1) == 15 )
12   {
13     v2 = 0;
14     if ( (*(a1 + 7) + *(a1 + 3) + *(a1 + 11)) % 7 == *(a1 + 14) % 7 )
15       v2 = 1;
16     if ( *a1 < *(a1 + 2) && *(a1 + 2) < *(a1 + 8) && *(a1 + 8) < *(a1 + 12) )
17       ++v2;
18     if ( *(a1 + 9) - *(a1 + 1) == 10 )
19       ++v2;
20     if ( sub_9F0(*(a1 + 5)) && sub_9F0(*(a1 + 13)) )
21       ++v2;
22     if ( (*(a1 + 10) + *(a1 + 8) + *(a1 + 7) + *(a1 + 6) + *(a1 + 5) + *(a1 + 4) + *(a1 + 13)) % 50 == 43 )
23       ++v2;
24     v3 = 0;
25     for ( i = 0; i < strlen(a1); ++i )
26     {
27       for ( j = i + 1; j < strlen(a1); ++j )
28       {
29         if ( sub_A2B(*(i + a1), *(j + a1)) != 1 )
30         {
31           v3 = 1;
32           break;
33         }
34       }
35     }
36     if ( !v3 )
37       ++v2;
38     v7 = 0;
39     for ( k = 0; k < strlen(a1); ++k )
40     {
41       if ( *(k + a1) > 96 && *(k + a1) <= 122 || *(k + a1) > 64 && *(k + a1) <= 90 || *(k + a1) > 47 && *(k + a1) <= 57 )
42         ++v7;
43     }
44     if ( v7 == strlen(a1) )
45       ++v2;
46     result = v2 == 7;
47   }
48   else
49   {
50     result = 0LL;
51   }
```

Terlihat bahwa panjang serial number yaitu 15 (line ke 11) lalu terdapat variable validasi v2 (line 13) yang nilainya akan bertambah bersamaan dengan pengecekan yang valid. Diakhir akan dicek bahwa v2 == 7 (line 46) artinya terdapat 7 pengecekan. Inputan serial kita sebut saja serials

1. serials[7] + serials[3] + serials[11]) % 7 == serials[14] % 7
2. serials[0] < serials[2] dan serials[2] < serials[8] dan serials[8] < serials[12]
3. serials[9] - serials[1] == 10
4. Terdapat fungsi sub_9F0 yang jika dibuka adalah sebuah fungsi primality test

```
1  signed __int64 __fastcall sub_9F0(signed int a1)
2  {
3    signed int i; // [sp+10h] [bp-4h]@1
4
5    for ( i = 2; i * i <= a1; ++i )
6    {
7      if ( !(a1 % i) )
8        return 0LL;
9    }
10   return 1LL;
11 }
```

Artinya serials[5] dan serials[13] haruslah bilangan prima.

5. (serials[10] + serials[8] + serials[7] + serials[6] + serials[5] + serials[4] + serials[13]) % 50 == 43
6. Terdapat sebuah looping pengecekan 2 angka ke suatu fungsi yang teridentifikasi adalah fungsi GCD (greatest common divisor)

```
int __fastcall sub_A2B(signed int a1, signed int a2)
{
  int result; // eax@2

  if ( a1 )
    result = sub_A2B((a2 % a1), a1);
  else
    result = a2;
  return result;
}
```

   Fungsi tersebut dipanggil pada nilai serials yang berpasangan ke samping kanan misal, gcd(1,2) gcd(1,3) ... (gcd 2,3) gcd(2,4) .. gcd(7,8) gcd(7,9)...
   Hasil dari pengecekan haruslah 1 artinya tidak ada pembagi bersama selain nilai 1.
7. Pengecekan bahwa serials haruslah sesuai dengan persamaan ini (serials[i] > 96 dan serials[i] <=122) atau (serials[i] > 64 dan serials[i] <= 90) atau (serials[i] > 47 dan serials[i] <= 57)

Terlihat bahwa ini merupakan problem SMT (Satisfiability modulo theories) yang dapat diselesaikan dengan algoritme solvernya salah satunya adalah Z3.

Namun terdapat pemanggilan fungsi loop (sub_9F0) dan rekursif (sub_A2B) yang tidak bisa langsung dijadikan constraint z3. Untuk penyelesaian aturan nomor 4 kita bisa generate terlebih dahulu angka prima yang mengikuti aturan nomor 7 kemudian buat contraint untuk serials[5] dan serials[13] nilai random dari bilangan prima tadi.

```
from z3 import *
import random

def is_prime(n):
    if n == 1:
        return False
    factors = 0
    for i in xrange(1, n+1):
        if n % i == 0:
            factors += 1
    if factors == 2:
        return True
    return False

if __name__ == "__main__":
        s = Solver()
        serials = [BitVec("ser_%d" % i, 32) for i in range(15)]
        my_primes = []
        for i in range(122):
```

```
                if ((i > 96 and  i <= 122) or (i > 64 and i <= 90) or (i > 47 and i <= 57))
and is_prime(i):
                        my_primes.append(i)
        s.add(serials[5] == random.choice(my_primes))
        s.add(serials[13] == random.choice(my_primes))
```

Aturan nomor 4 telah selesai, selanjutnya aturan nomor 6. Aturan nomor 6 lumayan sulit dipenuhi karena terdapat fungsi rekursif. Kita dapat membuat “linear” aturan ini agar dapat dijadikan constraint dengan cara generate angka prima dengan nilai kurang dari kemungkinan nilai gcd maksimum. Jika salah satu dari pasangan serials dapat dibagi oleh bilangan prima tersebut maka hasil gcd tidak mungkin bernilai 1. Contoh

```
gcd(98, 72) = 2 # tidak memenuhi aturan 6

Kita dapat mengetahuinya dengan cara seperti ini, kita tentukan bilangan primanya adalah 2

Jika X mod prima == 0 atau Y mod prima == 0 maka hasil gcd(X,Y) tidak mungkin 1

If 98 % 2 == 0 or 72 % 2 == 0 then False
>> False

98 % 2 == 0 # tidak memenuhi aturan 6 maka hasil gcd pasang angka tersebut tidak mungkin 1
```

Dapatkan kemungkinan nilai maksimum gcd

```
def gcd(a, b):
    if(a):
        res = gcd((b % a), a)
    else:
        res = b
    return res

max_gcd = 0
for i in range(1, 122, 1):
    for j in range(i+1, 122, 1):
        n = gcd(i, j)
        if n > max_gcd:
            max_gcd = n
print(max_gcd)

# >> 60
```

Didapatkan kemungkinan gcd maksimum yaitu 60, kita akan generate semua bilangan prima kurang dari 60, berikut scriptnya

```
max_gcd = 0
for i in range(1, 122, 1):
        for j in range(i+1, 122, 1):
                n = gcd(i, j)
```

```
                if n > max_gcd:
                        max_gcd = n

# max_gcd = 60
mod_primes = []
for i in range(max_gcd):
        if is_prime(i):
                mod_primes.append(i)

for i in range(15):
        for j in range(i+1, 15, 1):
                for k in mod_primes:
                        s.add(Or(serials[i] % k != 0, serials[j] % k != 0))
                s.add(serials[i] != serials[j])
```

Aturan 6 telah terpenuhi selanjutnya aturan 1, 2, 3, 5, dan 7 yang lumayan mudah

```
# aturan 1
s.add((serials[7] + serials[3] + serials[11]) % 7 == serials[14] % 7)

# aturan 2
s.add(And(serials[0] < serials[2], serials[2] < serials[8], serials[8] < serials[12]))

# aturan 3
s.add(serials[9] - serials[1] == 10)

# aturan 5
s.add((serials[10] + serials[8] + serials[7] + serials[6] + serials[5] + serials[4] +
serials[13]) % 50 == 43)

# aturan 7
for j in range(15):
        s.add(Or(And(serials[j] > 96, serials[j] <=122), And(serials[j] > 64, serials[j] <=
90), And(serials[j] > 47, serials[j] <= 57)))
```

Semua constraint telah dibuat lalu langkah terakhir yaitu menunggu z3 melakukan solving. Berikut scriptnya.

```
#!/usr/bin/env python
from z3 import *
import random

def gcd(a, b):
        if(a):
                res = gcd((b % a), a)
        else:
                res = b
        return res

def is_prime(n):
    if n == 1:
        return False
    factors = 0
    for i in xrange(1, n+1):
        if n % i == 0:
            factors += 1
    if factors == 2:
        return True
    return False
```

```
if __name__ == "__main__":
	s = Solver()
	serials = [BitVec("ser_%d" % i, 32) for i in range(15)] # definisikan constraint
variable sebesar 32 bit (4 byte)
	while True:
		my_primes = []
		for i in range(122):
			if ((i > 96 and  i <= 122) or (i > 64 and i <= 90) or (i > 47 and i <=
57)) and is_prime(i):
				my_primes.append(i)
		s.add(serials[5] == random.choice(my_primes))
		s.add(serials[13] == random.choice(my_primes))

		max_gcd = 0
		for i in range(1, 122, 1):
			for j in range(i+1, 122, 1):
				n = gcd(i, j)
				if n > max_gcd:
					max_gcd = n

		# max_gcd = 60

		mod_primes = []
		for i in range(max_gcd):
			if is_prime(i):
				mod_primes.append(i)

		for i in range(15):
			for j in range(i+1, 15, 1):
				for k in mod_primes:
					s.add(Or(serials[i] % k != 0, serials[j] % k != 0))
				s.add(serials[i] != serials[j])


		s.add((serials[7] + serials[3] + serials[11]) % 7 == serials[14] % 7)
		s.add(And(serials[0] < serials[2], serials[2] < serials[8], serials[8] <
serials[12]))
		s.add(serials[9] - serials[1] == 10)
		s.add((serials[10] + serials[8] + serials[7] + serials[6] + serials[5] +
serials[4] + serials[13]) % 50 == 43)
		for j in range(15):
			s.add(Or(And(serials[j] > 96, serials[j] <=122), And(serials[j] > 64,
serials[j] <= 90), And(serials[j] > 47, serials[j] <= 57)))

		if str(s.check()) == "sat":
			model = s.model()
			print "".join([chr(int(str(model[serials[i]]))) for i in range(15)])
		s.reset()
```

Didapatkan keygen yang valid:

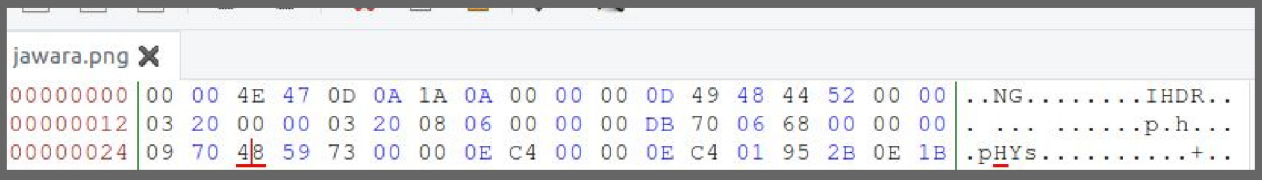
```
ACEGYkgIOMSaRme
2a5eqSC1Gk9OygY
3IARYq1CGSkyOa5
5kCaSqywsuYOtgG
5gR1ISGeoqaYykU
5YGLgSOAac1keCI
1O4USkey9Y5qgGC
DaGq9OsCIk15Sgy
0O15amkG7YgqIeS
1aIYceA5qkmCzgS
```

Picture (Forensics)

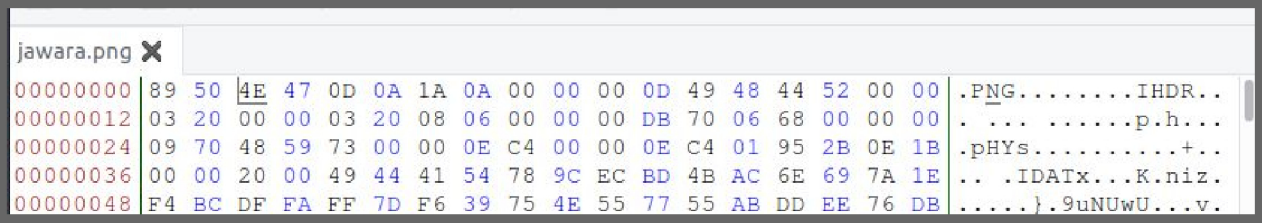
File:

```
curl -Ls https://git.io/vdnqC | base64 -d > pictures.7z
```

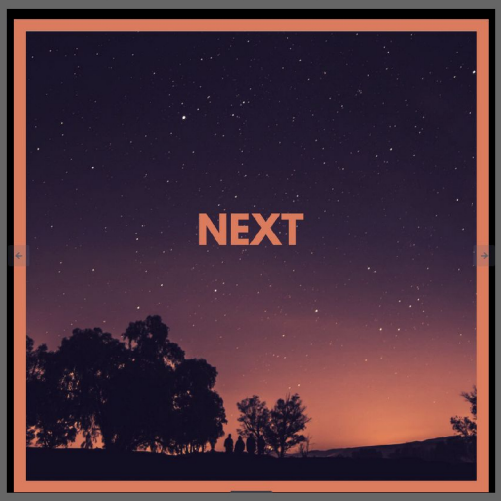
Diberikan file 7zip, lalu diekstrak didapatkan jawara.png. File tersebut tidak memiliki 2 byte header pertama, maka kami perbaiki sesuai dengan magic bytes[4]nya.



Setelah perbaikan.



Isi dari jawara.png ada gambar ini



Selanjutnya kami coba binwalk untuk melihat file apa saja yang ada didalam gambar tersebut.

```
$ binwalk -e jawara.png
```

[4] https://en.wikipedia.org/wiki/List_of_file_signatures

Didapatkan file .zllib, lalu kami ekstrak didapatkan file jpeg. Header file jpeg diperbaiki dengan cara yang sama seperti diatas lalu didapatkan gambar berikut ini:

Kami strings file jpeg tersebut didapatkan potongan flag lainnya

```
$ strings FLAG\ ISsss.jpg | grep flag
flag 1 : {m4r1_
flag3:HIJRAH}
```

Flag: CJ2017{m4r1_b3333333333e333r_HIJRAH}

RAM (Forensics)

File:

https://drive.google.com/open?id=0B_kAnrNjT4xeRDNYSDZObUQwLXM

Sebenarnya challenge ini dapat diselesaikan dengan strings karena mungkin problem setternya belom antisipasi cara ini.

```
$ strings memdump.mem | grep CJ2017
CJ2017{pertama_dan_terakhir}
```

Determinasi profile image tersebut dengan volatility

```
$ vol.py -f memdump.mem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG
search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86,
Win7SP1x86
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace
(/home/hrdn/Documents/rev/ram/memdump.mem)
                      PAE type : No PAE
                           DTB : 0x185000L
                          KDBG : 0x82927c28L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82928c00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2017-09-22 16:09:43 UTC+0000
     Image local date and time : 2017-09-22 09:09:43 -0700
```

Melihat proses yang sedang berjalan

```
$ vol.py -f memdump.mem --profile Win7SP1x86_23418 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                         Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x84195020 System                    4      0     82      517 ------      0 2017-09-23 06:05:23 UTC+0000
```

```
0x847db3c0 smss.exe            248     4     2      29 ------      0 2017-09-23 06:05:24 UTC+0000
0x84e62530 csrss.exe           316   308     9     365      0      0 2017-09-23 06:05:30 UTC+0000
0x84e82530 csrss.exe           364   356     9     293      1      0 2017-09-23 06:05:32 UTC+0000
0x84e8a530 wininit.exe         372   308     3      77      0      0 2017-09-23 06:05:32 UTC+0000
0x84e96530 winlogon.exe        400   356     3     113      1      0 2017-09-23 06:05:33 UTC+0000
0x84f05148 services.exe        460   372     8     192      0      0 2017-09-23 06:05:34 UTC+0000
0x84f0a408 lsass.exe           468   372     6     588      0      0 2017-09-23 06:05:34 UTC+0000
0x84f0e030 lsm.exe             476   372    10     145      0      0 2017-09-23 06:05:34 UTC+0000
0x84f2c030 svchost.exe         568   460    10     362      0      0 2017-09-23 06:05:39 UTC+0000
0x84f41b18 VBoxService.ex      628   460    12     115      0      0 2017-09-23 06:05:40 UTC+0000
0x847d5278 svchost.exe         692   460     7     258      0      0 2017-09-22 16:05:41 UTC+0000
0x84f7aa58 svchost.exe         780   460    19     478      0      0 2017-09-22 16:05:42 UTC+0000
0x84f884f0 svchost.exe         816   460    18     378      0      0 2017-09-22 16:05:42 UTC+0000
0x84f8dc30 svchost.exe         840   460    36     997      0      0 2017-09-22 16:05:42 UTC+0000
0x84f9e190 audiodg.exe         924   780     4     124      0      0 2017-09-22 16:05:44 UTC+0000
0x84fbc030 svchost.exe        1008   460    11     278      0      0 2017-09-22 16:05:46 UTC+0000
0x84e07030 svchost.exe        1088   460    13     368      0      0 2017-09-22 16:05:47 UTC+0000
0x84e13b90 spoolsv.exe        1224   460    12     279      0      0 2017-09-22 16:05:50 UTC+0000
0x84e34b18 svchost.exe        1276   460    17     316      0      0 2017-09-22 16:05:51 UTC+0000
0x84e8dd40 svchost.exe        1432   460    10     174      0      0 2017-09-22 16:05:53 UTC+0000
0x84eabb58 taskhost.exe       1456   460     8     181      1      0 2017-09-22 16:05:54 UTC+0000
0x850a1310 sppsvc.exe         1912   460     4     144      0      0 2017-09-22 16:06:02 UTC+0000
0x84f7d030 dwm.exe            1252   816     3      68      1      0 2017-09-22 16:06:17 UTC+0000
0x84851a48 explorer.exe       1348  1108    31     925      1      0 2017-09-22 16:06:18 UTC+0000
0x8514e620 VBoxTray.exe       1704  1348    13     150      1      0 2017-09-22 16:06:23 UTC+0000
0x85158690 Trojan.exe         1716  1348     5     145      1      0 2017-09-22 16:06:23 UTC+0000
0x85196030 SearchIndexer.      584   460    14     663      0      0 2017-09-22 16:06:30 UTC+0000
0x851b1ac0 dllhost.exe        1340   568    10     258      1      0 2017-09-22 16:06:33 UTC+0000
0x851f50a0 SearchProtocol     2092   584     7     322      0      0 2017-09-22 16:06:39 UTC+0000
0x8521e920 netsh.exe          2208  1716     0 --------      1      0 2017-09-22 16:06:43 UTC+0000   2017-09-22 16:06:46 UTC+0000
0x8511d460 mspaint.exe        2820  1348    10     336      1      0 2017-09-22 16:07:17 UTC+0000
0x8522d6f0 svchost.exe        2864   460     7      99      0      0 2017-09-22 16:07:19 UTC+0000
0x84e36030 svchost.exe        3444   460    13     356      0      0 2017-09-22 16:08:02 UTC+0000
0x84273400 FTK Imager.exe      420  1348    10     303      1      0 2017-09-22 16:08:49 UTC+0000
0x8425b638 notepad.exe        2984  1348     6     294      1      0 2017-09-22 16:08:55 UTC+0000
0x8502c030 SearchFilterHo     3304   584     5      99      0      0 2017-09-22 16:09:39 UTC+0000
0x84289b48 WmiPrvSE.exe       3512   568     8 175...74      0      0 2017-09-22 16:10:02 UTC+0000
```

Memdump Notepad

```
$ vol.py -f memdump.mem --profile Win7SP1x86_23418 memdump
--dump-dir=dump -p 2984
$ strings 2984.dmp  | grep CJ2017
CJ2017{pertama_dan_terakhir}
```

Flag didapatkan

Disc Forensic (Forensic)

File:

```
curl -L https://turl.ca/pzpmdn | base64 -d > disc.tar.gz (28 MB)
```

Terdapat 2 buah file yaitu file nha-13.vhd dan usbDrive.001. Kita foremost terlebih dahulu file usbDrive.001, didapatkan file rar. Namun file rar tersebut tidak bisa dibuka karena corrupt. Solusinya tambahkan saja flag “-kb” untuk “keep broken extracted files”

```
$ rar -kb e 00000400.rar
```

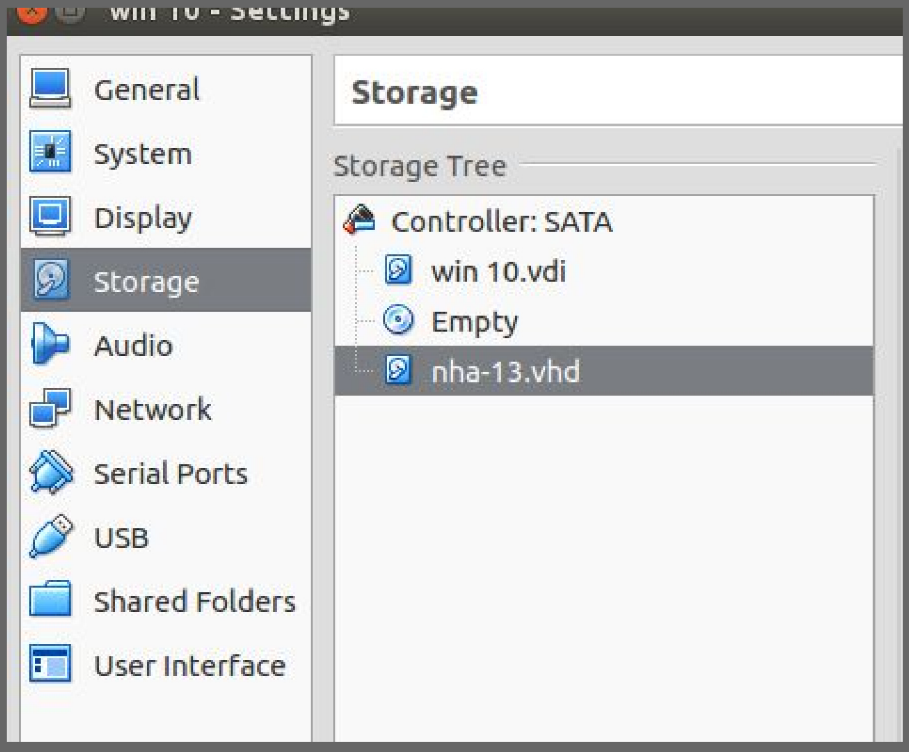
Didapatkan file text seperti ini

```
��BitLocker Drive EncrypBsyrinc occ

Ty rinpfc yhat yhps ps yhivyrBstBsyrinc occ,vymraBs yhivstaBtyf
yhivfyllywig pdinypfickewiyh yhivpdinypfickevaluivdisrlayid yukePC.

Idinypfick:

      6r35rA87-Br33-440F-BCE2-117B374248r9

If yhivabyrivpdinypfickematchis yhivivdisrlayid bc yukePC yhiusBs
yhivfyllywig occ tounltLo yuked Dri.

RBsyrinc Kcc:

      083985-489665-059873-228602-148467-207625-055044-445049

If yhivabyrivpdinypfickedyisn'tematch yhivivdisrlayid bc yukePC
yhiyhps psn'teyhiv Dght occ tounltLo yuked Dri.
Tnc anoyhirBsyrinc occvrBsfin
tohttrs://go.miEnosyft.ym/fwlik/?LikIr=260589vfyr addiypal
assistanci.
```

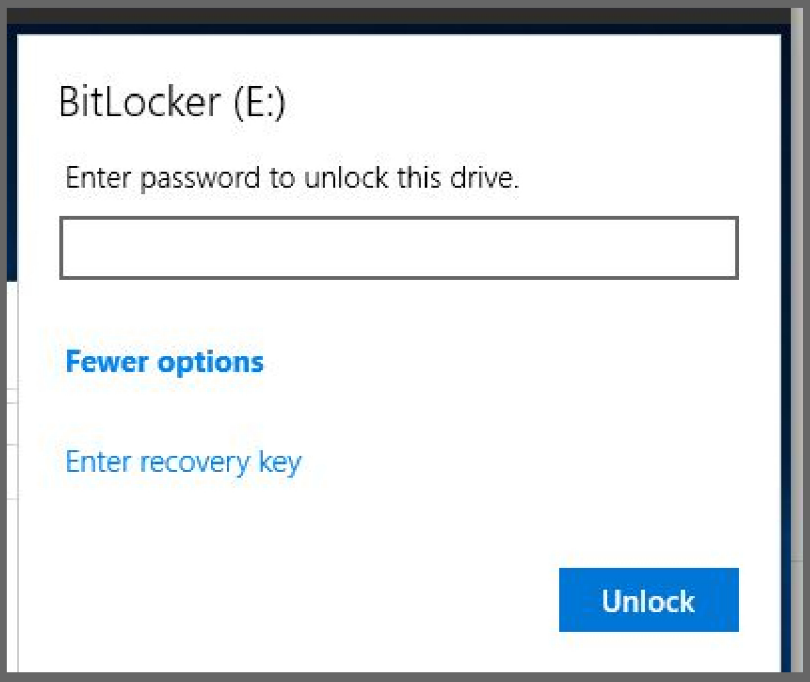
Sepertinya rusak karena recovery, namun recovery keynya masih terlihat

```
083985-489665-059873-228602-148467-207625-055044-445049
```

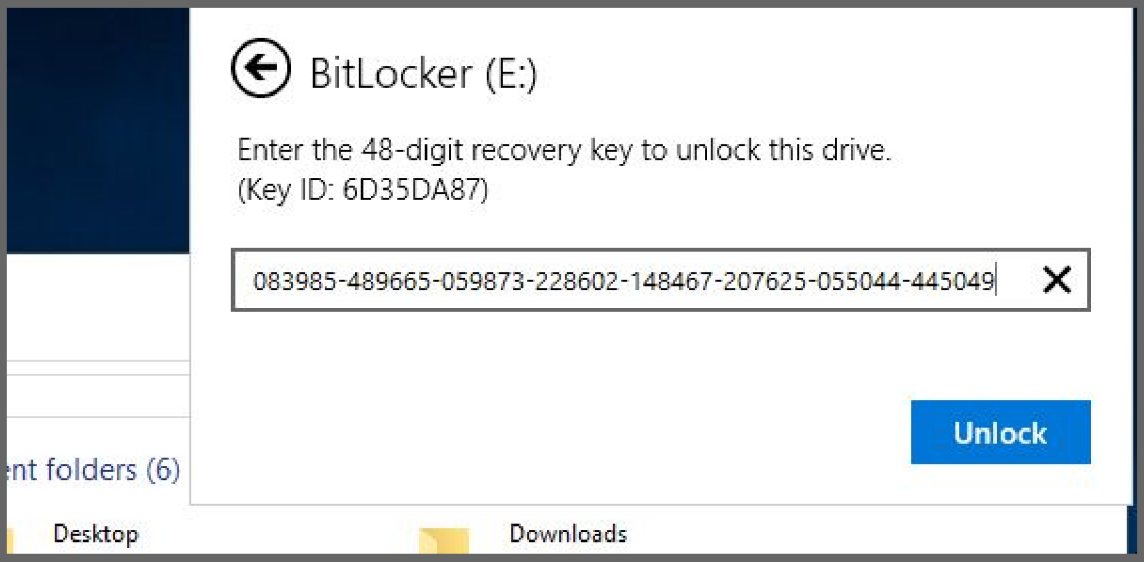
Load storange nha-13.vhd ke virtualbox



Buka drivenya



Masukan recovery keynya



Terdapat 4 file rar yang isinya database percakapan whatsapp. Decrypt dengan whatcrypt.com. Karena hintnya CJ{62..45} maka cari nomer hape yang sesuai dengan hint tersebut.